

## ЛАБОРАТОРНА РОБОТА №2

### ПОРІВНЯННЯ МЕТОДІВ КЛАСИФІКАЦІЇ ДАНИХ

**Мета роботи:** використовуючи спеціалізовані бібліотеки та мову програмування Python дослідити різні методи класифікації даних та навчитися їх порівнювати.

Хід роботи:

Завдання 2.1.

Класифікація за допомогою машин опорних векторів (SVM) Код програми:

```
1 import numpy as np
2 from sklearn import preprocessing
3 from sklearn.svm import LinearSVC
4 from sklearn.multiclass import OneVsOneClassifier
5 from sklearn.model_selection import train_test_split
6 from sklearn.model_selection import cross_val_score
7 input_file = "income_data.txt"
8 X = []
9 Y = []
10 count_class1 = 0
11 count_class2 = 0
12 max_datapoints = 25000
13 with open(input_file, "r") as f:
14     for line in f.readlines():
15         if count_class1 >= max_datapoints and count_class2 >= max_datapoints:
16             break
17         if '?' in line:
18             continue
19         data = line[:-1].split(',')
20         if data[-1] == '<=50K' and count_class1 < max_datapoints:
21             X.append(data)
22             count_class1 += 1
23         if data[-1] == '>50K' and count_class2 < max_datapoints:
24             X.append(data)
25             count_class2 += 1
26 X = np.array(X)
27 label_encoder = []
28 X_encoded = np.empty(X.shape)
29 for i, item in enumerate(X[0]):
```

					ДУ «Житомирська політехніка».20.121.12.						
Змн.	Арк.	№ докум.	Підпис	Дата				Літ.	Арк.	Аркуші	
Розроб.	Надворний М.Ю.										
Перевір.	Філіпов В.О.								1	19	
Керівник								ФІКТ Гр. ІПЗк-20-1			
Н. контр.											
Зав. каф.											

```

X_encoded = np.empty(X.shape)
for i, item in enumerate(X[0]):
    if item.isdigit():
        X_encoded[:, i] = X[:, i]
    else:
        label_encoder.append(preprocessing.LabelEncoder())
        X_encoded[:, i] = label_encoder[-1].fit_transform(X[:, i])
X = X_encoded[:, :-1].astype(int)
Y = X_encoded[:, -1].astype(int)
scaller = preprocessing.MinMaxScaler(feature_range=(0, 1))
X = scaller.fit_transform(X)
classifier = OneVsOneClassifier(LinearSVC(random_state=0))
classifier.fit(X=X, y=Y)
X_train, X_test, y_train, y_test \
= train_test_split(X, Y, test_size=0.2, random_state=5)
scaller = preprocessing.MinMaxScaler(feature_range=(0, 1))
X_train = scaller.fit_transform(X_train)
classifier.fit(X=X_train, y=y_train)
y_test_pred = classifier.predict(X_test)
f1 = cross_val_score(classifier, X, Y, scoring="f1_weighted", cv=3)
accuracy_values = cross_val_score(classifier, X, Y, scoring='accuracy', cv=3)
print("Accuracy: " + str(round(100 * accuracy_values.mean(), 2)) + "%")
precision_values = cross_val_score(classifier, X, Y, scoring='precision_weighted', cv=3)
print("Precision: " + str(round(100 * precision_values.mean(), 2)) + "%")
recall_values = cross_val_score(classifier, X, Y, scoring='recall_weighted', cv=3)
print("Recall: " + str(round(100 * recall_values.mean(), 2)) + "%")
f1_values = cross_val_score(classifier, X, Y, scoring='f1_weighted', cv=3)
print("F1: " + str(round(100 * f1_values.mean(), 2)) + "%")
print("F1 score: " + str(round(100 * f1.mean(), 2)) + "%")

print("F1 score: " + str(round(100 * f1.mean(), 2)) + "%")
input_data = ['37', 'Private', '215646', 'HS-grad', '9', 'Never-married', 'Handlers-cleaners',
'Not-in-family', 'White', 'Male', '0', '0', '40', 'United-States']
input_data_encoded = np.array([-1] * len(input_data))
count = 0
for i, item in enumerate(input_data):
    if item.isdigit():
        input_data_encoded[i] = item
    else:
        input_data_encoded[i] = int(label_encoder[count].transform([item]))
        count += 1
input_data_encoded = input_data_encoded.astype(int)
input_data_encoded = [input_data_encoded]
predicate_class = classifier.predict(input_data_encoded)
print(label_encoder[-1].inverse_transform(predicate_class)[0])

```

Рис 2.1 Код файлу LR\_2\_task\_1.py

Результат:

		Надворний М.Ю			ДУ «Житомирська політехніка».20.121.12 – Лр2	Арк.
		Філіпов В.О.				2
Змн.	Арк.	№ докум.	Підпис	Дата		

```

"D:\Штучний інтелект\Lab1\lab2\venv\Scripts\python.exe" "D:
Accuracy: 81.95%
Precision: 80.94%
Recall: 81.95%
F1: 80.13%
F1 score: 80.13%
>50K

Process finished with exit code 0

```

Рис 2.2 Результат програми файлу LR\_2\_task\_1.py

## Завдання 2.2.

Порівняння якості класифікаторів SVM з нелінійними ядрами

Код програми:

```

1 import numpy as np
2 from sklearn import preprocessing
3 from sklearn.svm import SVC
4 from sklearn.multiclass import OneVsOneClassifier
5 from sklearn.model_selection import train_test_split
6 from sklearn.model_selection import cross_val_score
7 input_file = "income_data.txt"
8 X = []
9 Y = []
10 count_class1 = 0
11 count_class2 = 0
12 max_datapoints = 25000
13 with open(input_file, "r") as f:
14     for line in f.readlines():
15         if count_class1 >= max_datapoints and count_class2 >= max_datapoints:
16             break
17         if '?' in line:
18             continue
19         data = line[:-1].split(',')
20         if data[-1] == '<=50K' and count_class1 < max_datapoints:
21             X.append(data)
22             count_class1 += 1
23         if data[-1] == '>50K' and count_class2 < max_datapoints:
24             X.append(data)
25             count_class2 += 1
26 X = np.array(X)
27 label_encoder = []
28 X_encoded = np.empty(X.shape)
29 for i, item in enumerate(X[0]):
30     if item.isdigit():
31         Y_encoded[i] = Y[i]

```

		Надворний М.Ю			ДУ «Житомирська політехніка».20.121.12 – Лр2	Арк.
		Філіпов В.О.				3
Змн.	Арк.	№ докум.	Підпис	Дата		

```

X_encoded = np.empty(X.shape)
for i, item in enumerate(X[0]):
    if item.isdigit():
        X_encoded[:, i] = X[:, i]
    else:
        label_encoder.append(preprocessing.LabelEncoder())
        X_encoded[:, i] = label_encoder[-1].fit_transform(X[:, i])
X = X_encoded[:, :-1].astype(int)
Y = X_encoded[:, -1].astype(int)
scaler = preprocessing.MinMaxScaler(feature_range=(0, 1))
X = scaler.fit_transform(X)
classifier = SVC(kernel='poly', degree=8)
classifier.fit(X=X, y=Y)
X_train, X_test, y_train, y_test \
= train_test_split(X, Y, test_size=0.2, random_state=5)
scaler = preprocessing.MinMaxScaler(feature_range=(0, 1))
X_train = scaler.fit_transform(X_train)
classifier.fit(X=X_train, y=y_train)
y_test_pred = classifier.predict(X_test)
f1 = cross_val_score(classifier, X, Y, scoring='f1_weighted', cv=3)
accuracy_values = cross_val_score(classifier, X, Y, scoring='accuracy', cv=3)
print("Accuracy: " + str(round(100 * accuracy_values.mean(), 2)) + "%")
precision_values = cross_val_score(classifier, X, Y, scoring='precision_weighted', cv=3)
print("Precision: " + str(round(100 * precision_values.mean(), 2)) + "%")
recall_values = cross_val_score(classifier, X, Y, scoring='recall_weighted', cv=3)
print("Recall: " + str(round(100 * recall_values.mean(), 2)) + "%")
f1_values = cross_val_score(classifier, X, Y, scoring='f1_weighted', cv=3)
print("F1: " + str(round(100 * f1_values.mean(), 2)) + "%")
print("F1 score: " + str(round(100 * f1.mean(), 2)) + "%")
input_data = ['37', 'Private', '215646', 'HS-grad', '9', 'Never-married', 'Handlers-cleaners', 'Not-in-family', 'White', 'Male', '']

```

```

input_data_encoded = np.array([-1] * len(input_data))
count = 0
for i, item in enumerate(input_data):
    if item.isdigit():
        input_data_encoded[i] = item
    else:
        input_data_encoded[i] = int(label_encoder[count].transform([item]))
        count += 1
input_data_encoded = input_data_encoded.astype(int)
input_data_encoded = [input_data_encoded]
predicate_class = classifier.predict(input_data_encoded)
print(label_encoder[-1].inverse_transform(predicate_class)[0])

```

Рис 2.3 Код файлу LR\_2\_task\_2\_1.py

		Надворний М.Ю			ДУ «Житомирська політехніка».20.121.12 – Лр2	Арк.
		Філіпов В.О.				4
Змн.	Арк.	№ докум.	Підпис	Дата		

```

1 import numpy as np
2 from sklearn import preprocessing
3 from sklearn.svm import SVC
4 from sklearn.multiclass import OneVsOneClassifier
5 from sklearn.model_selection import train_test_split
6 from sklearn.model_selection import cross_val_score
7 input_file = "income_data.txt"
8 X = []
9 Y = []
10 count_class1 = 0
11 count_class2 = 0
12 max_datapoints = 25000
13 with open(input_file, "r") as f:
14     for line in f.readlines():
15         if count_class1 >= max_datapoints and count_class2 >= max_datapoints:
16             break
17         if '?' in line:
18             continue
19         data = line[:-1].split(' ')
20         if data[-1] == '<=50K' and count_class1 < max_datapoints:
21             X.append(data)
22             count_class1 += 1
23         if data[-1] == '>50K' and count_class2 < max_datapoints:
24             X.append(data)
25             count_class2 += 1
26 X = np.array(X)
27 label_encoder = []
28 X_encoded = np.empty(X.shape)
29 for i, item in enumerate(X[0]):
30     if item.isdigit():
31         X_encoded[:, i] = Y[:, i]
32     else:
33         label_encoder.append(preprocessing.LabelEncoder())
34         X_encoded[:, i] = label_encoder[-1].fit_transform(X[:, i])
35 X = X_encoded[:, :-1].astype(int)
36 Y = X_encoded[:, -1].astype(int)
37 scaler = preprocessing.MinMaxScaler(feature_range=(0, 1))
38 X = scaler.fit_transform(X)
39 classifier = SVC(kernel='rbf')
40 classifier.fit(X=X, y=Y)
41 X_train, X_test, y_train, y_test = \
42     train_test_split(X, Y, test_size=0.2, random_state=5)
43 scaler = preprocessing.MinMaxScaler(feature_range=(0, 1))
44 X_train = scaler.fit_transform(X_train)
45 classifier.fit(X=X_train, y=y_train)
46 y_test_pred = classifier.predict(X_test)
47 f1 = cross_val_score(classifier, X, Y, scoring='f1_weighted', cv=3)
48 accuracy_values = cross_val_score(classifier, X, Y, scoring='accuracy', cv=3)
49 print("Accuracy: " + str(round(100 * accuracy_values.mean(), 2)) + "%")
50 precision_values = cross_val_score(classifier, X, Y, scoring='precision_weighted', cv=3)
51 print("Precision: " + str(round(100 * precision_values.mean(), 2)) + "%")
52 recall_values = cross_val_score(classifier, X, Y, scoring='recall_weighted', cv=3)
53 print("Recall: " + str(round(100 * recall_values.mean(), 2)) + "%")
54 f1_values = cross_val_score(classifier, X, Y, scoring='f1_weighted', cv=3)
55 print("F1: " + str(round(100 * f1_values.mean(), 2)) + "%")
56 print("F1 score: " + str(round(100 * f1.mean(), 2)) + "%")
57 input_data = ['37', 'Private', '215646', 'HS-grad', '9', 'Never-married', 'Handlers-cleaners',
58             'Not-in-family', 'White', 'Male', '0', '0', '40', 'United-States']
59 input_data_encoded = np.array([-1] * len(input_data))
60 count = 0

```

		Надворний М.Ю			ДУ «Житомирська політехніка».20.121.12 – Лр2	Арк.
		Філіпов В.О.				5
Змн.	Арк.	№ докум.	Підпис	Дата		

```

for i, item in enumerate(input_data):
    if item.isdigit():
        input_data_encoded[i] = item
    else:
        input_data_encoded[i] = int(label_encoder[count].transform([item]))
        count += 1
input_data_encoded = input_data_encoded.astype(int)
input_data_encoded = [input_data_encoded]
predicate_class = classifier.predict(input_data_encoded)
print(label_encoder[-1].inverse_transform(predicate_class)[0])

```

Рис 2.4 Код файла LR\_2\_task\_2\_2.py

```

import numpy as np
from sklearn import preprocessing
from sklearn.svm import SVC
from sklearn.multiclass import OneVsOneClassifier
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
input_file = "income_data.txt"
X = []
Y = []
count_class1 = 0
count_class2 = 0
max_datapoints = 25000
with open(input_file, "r") as f:
    for line in f.readlines():
        if count_class1 >= max_datapoints and count_class2 >= max_datapoints:
            break
        if '?' in line:
            continue
        data = line[:-1].split(',')
        if data[-1] == '<=50K' and count_class1 < max_datapoints:
            X.append(data)
            count_class1 += 1
        if data[-1] == '>50K' and count_class2 < max_datapoints:
            X.append(data)
            count_class2 += 1
X = np.array(X)
label_encoder = []
X_encoded = np.empty(X.shape)
for i, item in enumerate(X[0]):
    if item.isdigit():
        X_encoded[i] = item

```

		Надворний М.Ю			ДУ «Житомирська політехніка».20.121.12 – Лр2	Арк.
		Філіпов В.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		6

```

X_encoded[:, i] = X[:, i]
else:
    label_encoder.append(preprocessing.LabelEncoder())
    X_encoded[:, i] = label_encoder[-1].fit_transform(X[:, i])
X = X_encoded[:, :-1].astype(int)
Y = X_encoded[:, -1].astype(int)
scaller = preprocessing.MinMaxScaler(feature_range=(0, 1))
X = scaller.fit_transform(X)
classifier = SVC(kernel='sigmoid')
classifier.fit(X=X, y=Y)
X_train, X_test, y_train, y_test = \
train_test_split(X, Y, test_size=0.2, random_state=5)
scaller = preprocessing.MinMaxScaler(feature_range=(0, 1))
X_train = scaller.fit_transform(X_train)
classifier.fit(X=X_train, y=y_train)
y_test_pred = classifier.predict(X_test)
f1 = cross_val_score(classifier, X, Y, scoring='f1_weighted', cv=3)
accuracy_values = cross_val_score(classifier, X, Y, scoring='accuracy', cv=3)
print("Accuracy: " + str(round(100 * accuracy_values.mean(), 2)) + "%")
precision_values = cross_val_score(classifier, X, Y, scoring='precision_weighted', cv=3)
print("Precision: " + str(round(100 * precision_values.mean(), 2)) + "%")
recall_values = cross_val_score(classifier, X, Y, scoring='recall_weighted', cv=3)
print("Recall: " + str(round(100 * recall_values.mean(), 2)) + "%")
f1_values = cross_val_score(classifier, X, Y, scoring='f1_weighted', cv=3)
print("F1: " + str(round(100 * f1_values.mean(), 2)) + "%")
print("F1 score: " + str(round(100 * f1.mean(), 2)) + "%")
input_data = ['37', 'Private', '215646', 'HS-grad', '9', 'Never-married', 'Handlers-cleaners',
'Not-in-family', 'White', 'Male', '0', '0', '40', 'United-States']
input_data_encoded = np.array([-1] * len(input_data))
count = 0
for i, item in enumerate(input_data):
    if item.isdigit():
        input_data_encoded[i] = item
    else:
        input_data_encoded[i] = int(label_encoder[count].transform([item]))
        count += 1
input_data_encoded = input_data_encoded.astype(int)
input_data_encoded = [input_data_encoded]
predicate_class = classifier.predict(input_data_encoded)
print(label_encoder[-1].inverse_transform(predicate_class)[0])

```

Рис 2.5 Код файлу LR\_2\_task\_2\_3.py

Результат:

```

Accuracy: 83.99%
Precision: 83.21%
Recall: 83.99%
F1: 82.99%
F1 score: 82.99%
k=50K

```

Рис 2.6 Результат Поліноміального ядра

		Надворний М.Ю			ДУ «Житомирська політехніка».20.121.12 – Лр2	Арк.
		Філіпов В.О.				7
Змн.	Арк.	№ докум.	Підпис	Дата		

```

Accuracy: 83.96%
Precision: 83.18%
Recall: 83.96%
F1: 82.95%
F1 score: 82.95%
<=50K

```

Рис 2.7 Результат гаусового ядра

```

Accuracy: 57.26%
Precision: 57.1%
Recall: 57.26%
F1: 57.18%
F1 score: 57.18%
<=50K

```

Рис 2.8 Результат сигмоїдального ядра

Висновок: в даній ситуації краще за всього справляється RBF, має кращу точність та швидкість. Сигмоїдне ядро не так добре, так як відстає по швидкості.

### Завдання 2.3

Порівняння якості класифікаторів на прикладі класифікації сортів ірисів

Код програми:

		Надворний М.Ю			ДУ «Житомирська політехніка».20.121.12 – Лр2	Арк.
		Філіпов В.О.				8
Змн.	Арк.	№ докум.	Підпис	Дата		



```

from sklearn.datasets import load_iris
import numpy as np
from pandas import read_csv
from pandas.plotting import scatter_matrix
from matplotlib import pyplot
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
iris_dataset = load_iris()
print("Ключі iris dataset : \n{}".format(iris_dataset.keys()))
print(iris_dataset["DESCR"][:193] + "\n...")
print("Назви відповідей: {}".format(iris_dataset["target_names"]))

print("Назви ознак: \n{}".format(iris_dataset["feature_names"]))
print("Тип масиву data: {}".format(type(iris_dataset["data"])))
print("Форма масиву data: {}".format(iris_dataset["data"].shape))
print("Тип масиву target: {}".format(type(iris_dataset['target'])))
print("Відповіді:\n{}".format(iris_dataset['target']))
url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/iris.csv"
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width', 'class']

```

```

# shape
print(dataset.shape)
# Зріз даних head
print(dataset.head(20))
# Статистичні зведення методом describe
print(dataset.describe())
# Розподіл за атрибутом class
print(dataset.groupby('class').size())
# Діаграма розмаху
dataset.plot(kind='box', subplots=True, layout=(2, 2), sharex=False, sharey=False)
pyplot.show()
# Гістограма розподілу атрибутів датасета
dataset.hist()
pyplot.show()
# Матриця діаграм розсіювання
scatter_matrix(dataset)
pyplot.show()
# Розділення датасету на навчальну та контрольну вибірки
array = dataset.values
# Вибір перших 4-х стовпців
X = array[:, 0:4]
# Вибір 5-го стовпця
y = array[:, 4]
# Розділення X и y на навчальну та контрольну вибірки
X_train, X_validation, Y_train, Y_validation = train_test_split(X, y, test_size=0.20,
random_state=1)
models = []
models.append(('LR', LogisticRegression(solver='liblinear', multi_class='ovr')))
models.append(('LDA', LinearDiscriminantAnalysis()))

```

		Надворний М.Ю			ДУ «Житомирська політехніка».20.121.12 – Лр2	Арк.
		Філіпов В.О.				9
Змн.	Арк.	№ докум.	Підпис	Дата		

```

models.append(('LR', LogisticRegression(solver='liblinear', multi_class='ovr')))
models.append(('LDA', LinearDiscriminantAnalysis()))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier()))
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC(gamma='auto')))
results = []
names = []
for name, model in models:
    kfold = StratifiedKFold(n_splits=10, random_state=1, shuffle=True)
    cv_results = cross_val_score(model, X_train, Y_train, cv=kfold, scoring='accuracy')
    results.append(cv_results)
    names.append(name)
    print('%s: %f (%f)' % (name, cv_results.mean(), cv_results.std()))
# Порівняння алгоритмів
pyplot.boxplot(results, labels=names)
pyplot.title('Algorithm Comparison')
pyplot.show()
# Створюємо прогноз на контрольний вибірку
model = SVC(gamma='auto')
model.fit(X_train, Y_train)
predictions = model.predict(X_validation)
# Оцінюємо прогноз
print(accuracy_score(Y_validation, predictions))
print(confusion_matrix(Y_validation, predictions))
print(classification_report(Y_validation, predictions))
X_new = np.array([[5, 2.9, 1, 0.2]])
for name, model in models:
    model.fit(X_train, Y_train)
    prediction = model.predict(X_new)
    print("Прогноз: {}".format(prediction))
    print(accuracy_score(Y_validation, predictions))
    print(confusion_matrix(Y_validation, predictions))
    print(classification_report(Y_validation, predictions))

```

Рис 2.9 Код файлу LR\_2\_task\_3.py

Результат:

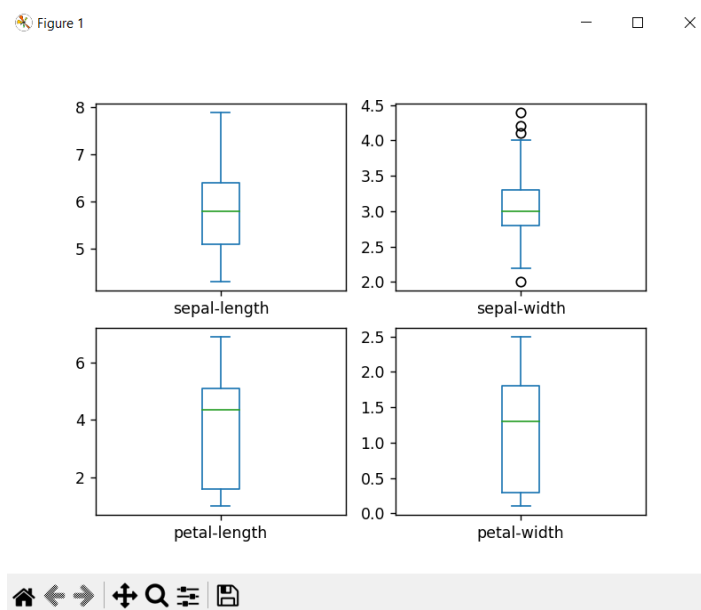


Рис. 2.10 Результат діаграми розмаху

		Надворний М.Ю			ДУ «Житомирська політехніка».20.121.12 – Лр2	Арк.
		Філіпов В.О.				10
Змн.	Арк.	№ докум.	Підпис	Дата		

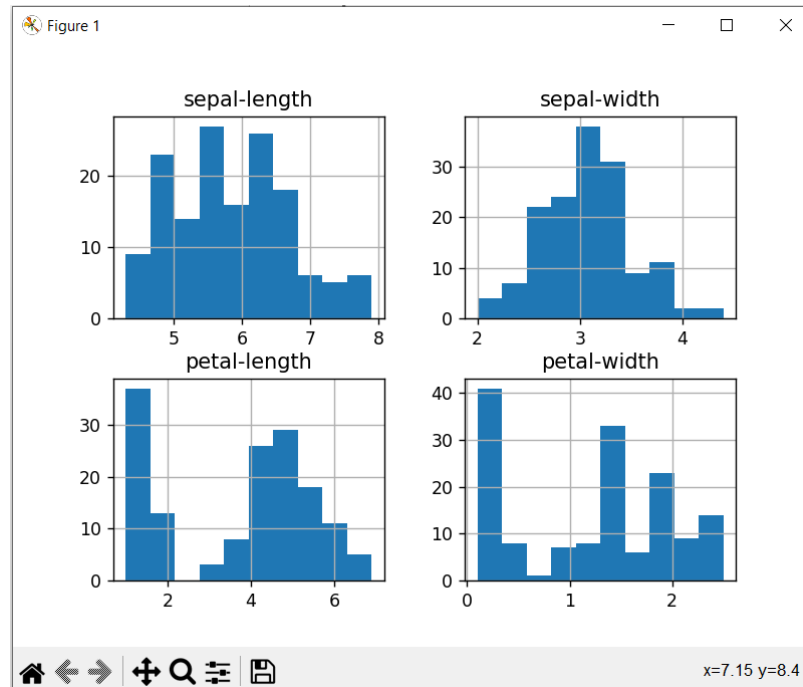


Рис. 2.11 Гістограма розподілу атрибутів

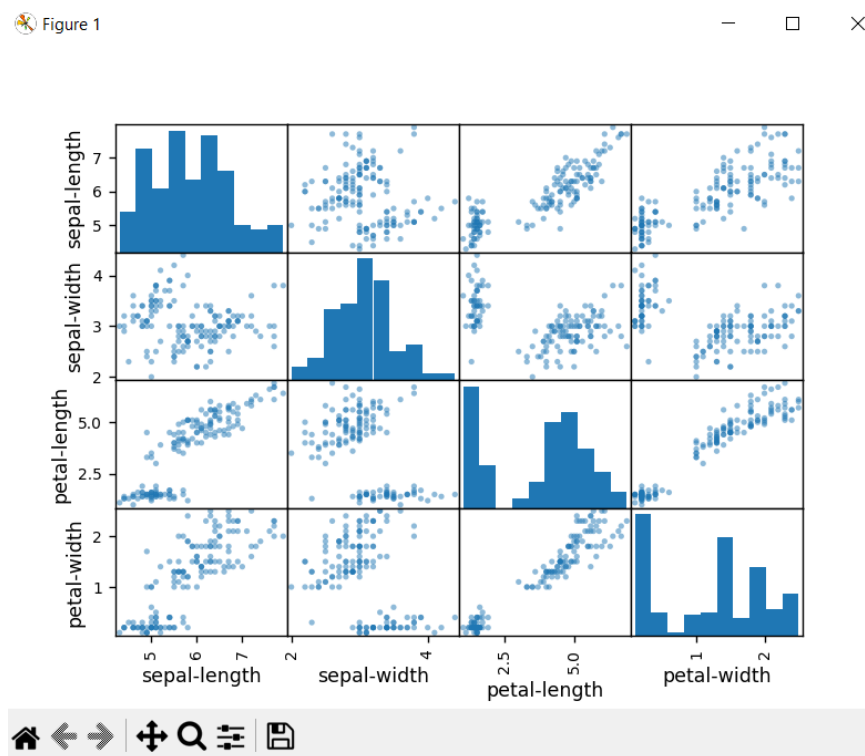


Рис. 2.12 Матриця діаграми розсіювання



```

8      4.4      2.9      1.4      0.2 Iris-setosa
9      4.9      3.1      1.5      0.1 Iris-setosa
10     5.4      3.7      1.5      0.2 Iris-setosa
11     4.8      3.4      1.6      0.2 Iris-setosa
12     4.8      3.0      1.4      0.1 Iris-setosa
13     4.3      3.0      1.1      0.1 Iris-setosa
14     5.8      4.0      1.2      0.2 Iris-setosa
15     5.7      4.4      1.5      0.4 Iris-setosa
16     5.4      3.9      1.3      0.4 Iris-setosa
17     5.1      3.5      1.4      0.3 Iris-setosa
18     5.7      3.8      1.7      0.3 Iris-setosa
19     5.1      3.8      1.5      0.3 Iris-setosa
count  150.000000  150.000000  150.000000  150.000000
mean    5.843333   3.054000   3.758667   1.198667
std     0.828066   0.433594   1.764420   0.763161
min     4.300000   2.000000   1.000000   0.100000
25%     5.100000   2.800000   1.600000   0.300000
50%     5.800000   3.000000   3.350000   1.300000
75%     6.400000   3.300000   5.100000   1.800000
max     7.900000   4.400000   6.900000   2.500000
class
Iris-setosa      50
Iris-versicolor  50
Iris-virginica   50
dtype: int64
LR: 0.941667 (0.065085)
LDA: 0.975000 (0.038188)
KNN: 0.958333 (0.041667)
CART: 0.958333 (0.041667)
NB: 0.950000 (0.05277)
SVM: 0.983333 (0.033333)

```

Рис. 2.14 Результат програми

Висновок: найкраще показала себе модель лінійного дискримінантного аналізу.  
Квітка належала до класу Iris-setosa.

#### Завдання 2.4.

Порівняння якості класифікаторів для набору даних завдання 2.1

Код програми:

```

import numpy as np
from sklearn import preprocessing
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.naive_bayes import GaussianNB

input_file = "income_data.txt"
X = []
Y = []
count_class1 = 0
count_class2 = 0
max_datapoints = 25000
with open(input_file, "r") as f:
    for line in f.readlines():
        if count_class1 >= max_datapoints and count_class2 >= max_datapoints:
            break
        if '?' in line:
            continue
        data = line[:-1].split(',')
        if data[-1] == '<=50K' and count_class1 < max_datapoints:
            X.append(data)
            count_class1 += 1
        if data[-1] == '>50K' and count_class2 < max_datapoints:
            X.append(data)

```

```

X = np.array(X)
label_encoder = []
X_encoded = np.empty(X.shape)
for i, item in enumerate(X[0]):
    if item.isdigit():
        X_encoded[:, i] = X[:, i]
    else:
        label_encoder.append(preprocessing.LabelEncoder())
        X_encoded[:, i] = label_encoder[-1].fit_transform(X[:, i])
X = X_encoded[:, :-1].astype(int)
Y = X_encoded[:, -1].astype(int)
scaller = preprocessing.MinMaxScaler(feature_range=(0, 1))
X = scaller.fit_transform(X)

classifier = SVC(gamma='auto')

classifier.fit(X=X, y=Y)
X_train, X_test, y_train, y_test \
    = train_test_split(X, Y, test_size=0.2, random_state=5)
scaller = preprocessing.MinMaxScaler(feature_range=(0, 1))
X_train = scaller.fit_transform(X_train)
classifier.fit(X=X_train, y=y_train)
y_test_pred = classifier.predict(X_test)
f1 = cross_val_score(classifier, X, Y, scoring="f1_weighted", cv=3)
accuracy_values = cross_val_score(classifier, X, Y, scoring='accuracy', cv=3)
print("Accuracy: " + str(round(100 * accuracy_values.mean(), 2)) + "%")
precision_values = cross_val_score(
    classifier, X, Y, scoring='precision_weighted', cv=3)
print("Precision: " + str(round(100 * precision_values.mean(), 2)) + "%")

```

		Надворний М.Ю			ДУ «Житомирська політехніка».20.121.12 – Лр2	Арк.
		Філіпов В.О.				14
Змн.	Арк.	№ докум.	Підпис	Дата		

```

print("Accuracy: " + str(round(100 * accuracy_values.mean(), 2)) + "%")
precision_values = cross_val_score(
    classifier, X, Y, scoring='precision_weighted', cv=3)
print("Precision: " + str(round(100 * precision_values.mean(), 2)) + "%")
recall_values = cross_val_score(
    classifier, X, Y, scoring='recall_weighted', cv=3)
print("Recall: " + str(round(100 * recall_values.mean(), 2)) + "%")
f1_values = cross_val_score(classifier, X, Y, scoring='f1_weighted', cv=3)
print("F1: " + str(round(100 * f1_values.mean(), 2)) + "%")
print("F1 score: " + str(round(100 * f1.mean(), 2)) + "%")
input_data = ['37', 'Private', '215646', 'HS-grad', '9', 'Never-married', 'Handlers-cleaners',
    'Not-in-family', 'White', 'Male', '0', '0', '40', 'United-States']
input_data_encoded = np.array([-1] * len(input_data))
count = 0
for i, item in enumerate(input_data):
    if item.isdigit():
        input_data_encoded[i] = item
    else:
        input_data_encoded[i] = int(label_encoder[count].transform([item]))
        count += 1
input_data_encoded = input_data_encoded.astype(int)
input_data_encoded = [input_data_encoded]
predicate_class = classifier.predict(input_data_encoded)
print(label_encoder[-1].inverse_transform(predicate_class)[0])

```

Рис 2.15 Код файлу LR\_2\_task\_4.py

Результат:

```

Accuracy: 81.82%
Precision: 80.69%
Recall: 81.82%
F1: 80.25%
F1 score: 80.25%
>50K

```

Рис.2.16 Точність класифікатора LR

```

Accuracy: 81.14%
Precision: 79.86%
Recall: 81.14%
F1: 79.35%
F1 score: 79.35%
>50K

```

Рис. 2.17 Точність класифікатора LDA

		Надворний М.Ю			ДУ «Житомирська політехніка».20.121.12 – Лр2	Арк.
		Філіпов В.О.				15
Змн.	Арк.	№ докум.	Підпис	Дата		

```

Accuracy: 82.16%
Precision: 81.53%
Recall: 82.16%
F1: 81.75%
F1 score: 81.75%
<=50K

```

Рис. 2.18 Точність класифікатора KNN

```

Accuracy: 80.55%
Precision: 80.76%
Recall: 80.66%
F1: 80.84%
F1 score: 80.77%
>50K

```

Рис. 2.19 Точність класифікатора CART

```

Accuracy: 79.76%
Precision: 78.2%
Recall: 79.76%
F1: 77.13%
F1 score: 77.13%
<=50K

```

Рис. 2.20 Точність класифікатора NB

```

Accuracy: 82.38%
Precision: 81.51%
Recall: 82.38%
F1: 80.6%
F1 score: 80.6%
>50K

```

Рис. 2.21 Точність класифікатора SVM

### Завдання 2.5.

Класифікація даних лінійним класифікатором Ridge

Код програми:

		Надворний М.Ю			ДУ «Житомирська політехніка».20.121.12 – Лр2	Арк.
		Філіпов В.О.				16
Змн.	Арк.	№ докум.	Підпис	Дата		



```

import numpy as np
import seaborn as sns
from sklearn.datasets import load_iris
from sklearn.linear_model import RidgeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from io import BytesIO
import matplotlib.pyplot as plt
from sklearn import metrics

sns.set()
iris = load_iris()
X, y = iris.data, iris.target
Xtrain, Xtest, ytrain, ytest = train_test_split(
    X, y, test_size=0.3, random_state=0)
clf = RidgeClassifier(tol=1e-2, solver="sag")
clf.fit(Xtrain, ytrain)
ypred = clf.predict(Xtest)
print('Accuracy:', np.round(metrics.accuracy_score(ytest, ypred), 4))
print('Precision:', np.round(metrics.precision_score(
    ytest, ypred, average='weighted'), 4))
print('Recall:', np.round(metrics.recall_score(
    ytest, ypred, average='weighted'), 4))
print('F1 Score:', np.round(metrics.f1_score(ytest, ypred, average='weighted'), 4))
print('Cohen Kappa Score:', np.round(
    metrics.cohen_kappa_score(ytest, ypred), 4))
print('Matthews Corrccoef:', np.round(
    metrics.matthews_corrcoef(ytest, ypred), 4))
print('\t\tClassification Report:\n',
    metrics.classification_report(ypred, ytest))

    metrics.classification_report(ypred, ytest))
mat = confusion_matrix(ytest, ypred)
sns.heatmap(mat.T, square=True, annot=True, fmt='d', cbar=False)
plt.xlabel('true label')
plt.ylabel('predicted label')
plt.savefig("Confusion.jpg")

f = BytesIO()
plt.savefig(f, format="svg")

```

Рис 2.22 Код файлу LR\_2\_task\_5.py

Результат:

```

Accuracy: 0.7556
Precision: 0.8333
Recall: 0.7556
F1 Score: 0.7503
Cohen Kappa Score: 0.6431
Matthews Corrccoef: 0.6831

```

		Надворний М.Ю			ДУ «Житомирська політехніка».20.121.12 – Лр2	Арк.
		Філіпов В.О.				17
Змн.	Арк.	№ докум.	Підпис	Дата		

Classification Report:				
	precision	recall	f1-score	support
0	1.00	1.00	1.00	16
1	0.44	0.89	0.59	9
2	0.91	0.50	0.65	20
accuracy			0.76	45
macro avg	0.78	0.80	0.75	45
weighted avg	0.85	0.76	0.76	45

Рис. 2.23 Результат програми

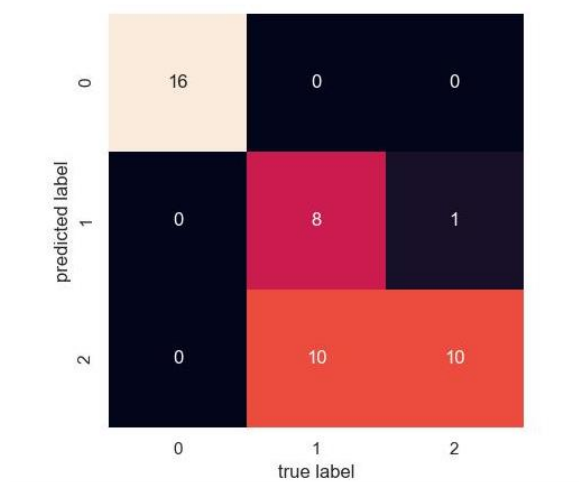


Рис. 2.24 Матриця невідповідності

Висновок: Матриця невідповідності – це таблиця особливого компонування, що дає можливість унаочнювати продуктивність алгоритму. Кожен з рядків цієї матриці представляє зразки прогнозованого класу, тоді як кожен зі стовпців представляє зразки справжнього класу (або навпаки).

Коефіцієнт каппа Коена статистика, яка використовується для вимірювання надійності між експертами для якісних пунктів.

Кореляції Метьюза – використовується в машинному навчанні, як міра якості бінарних мультикласних класифікацій.

Висновок: в ході виконання лабораторної роботи використовуючи спеціалізовані бібліотеки та мову програмування Python дослідив різні методи класифікації даних та навчився їх порівнювати.

GitHub: <https://github.com/Max1648/Artificial-Intelligence2> - новий гіт, з першим виникли проблеми.

Старий гіт - <https://github.com/Max1648/Artificial-Intelligence>

		Надворний М.Ю			ДУ «Житомирська політехніка».20.121.12 – Лр2	Арк.
		Філіпов В.О.				19
Змн.	Арк.	№ докум.	Підпис	Дата		